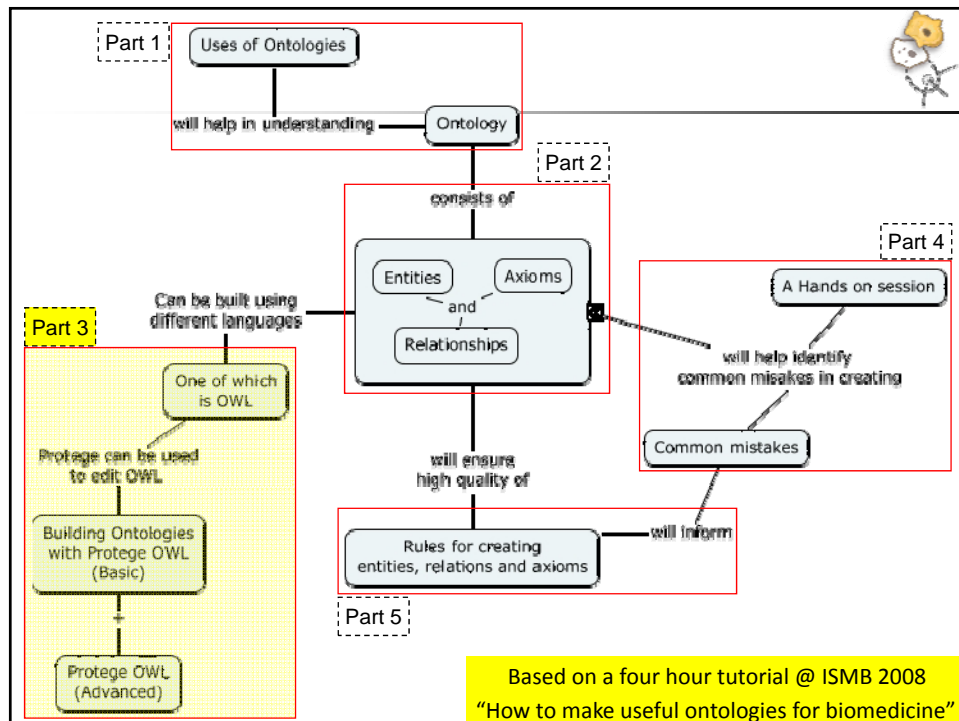


Developing Biomedical Ontologies using Protégé

(ICF Ontology and Protégé workshop 2008)

Nigam Shah

Stanford University
nigam@stanford.edu





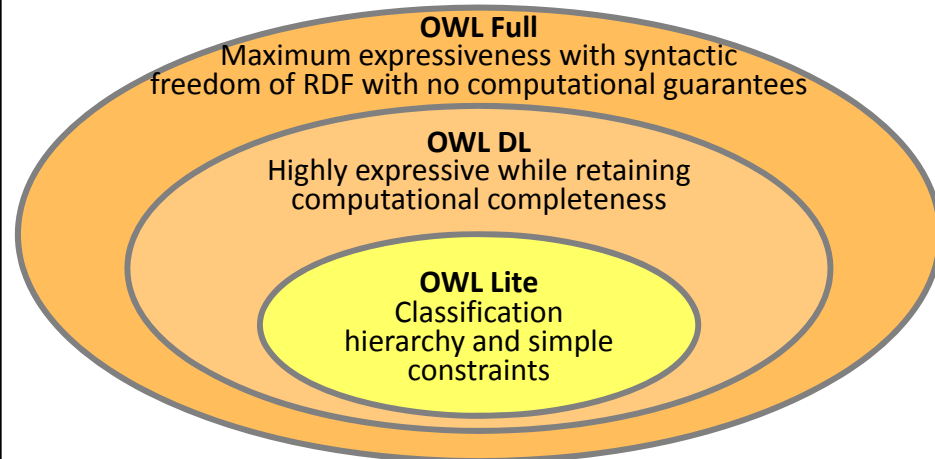
Overview of OWL

Nigam Shah
nigam@stanford.edu



OWL

- Web Ontology Language
- Recommended by W3C since Feb 2004
- Based on predecessors (DAML+OIL)
- A Web Language: Based on RDF(S)
- An Ontology Language: Based on logic
- Three varieties
 - OWL-full
 - OWL-DL (“OWL”)
 - OWL-Lite



OWL constructs

[illegible]

Working with OWL syntax is not easy

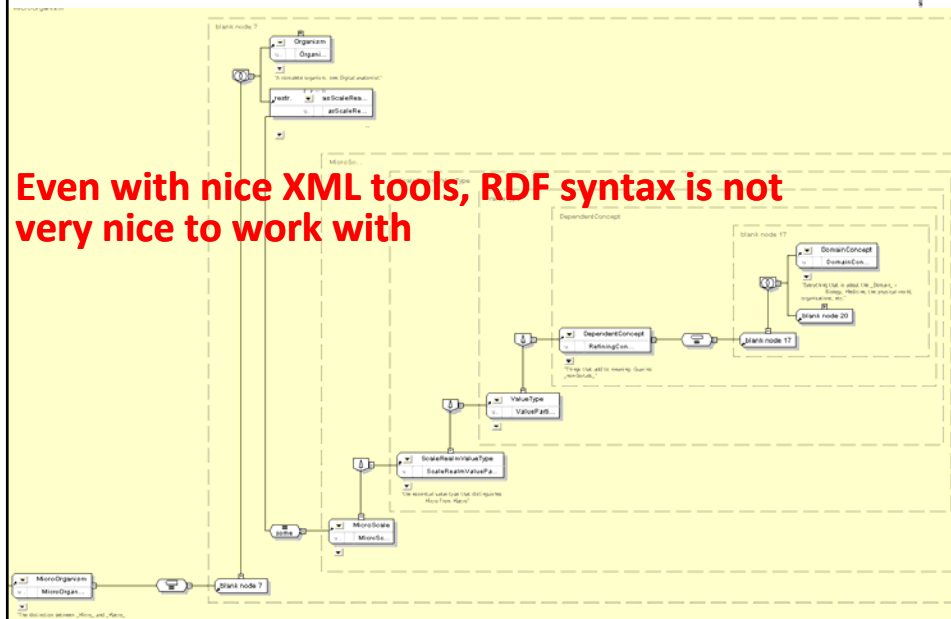


```
<owl:Class rdf:ID="Virus">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  </rdfs:comment>
  <owl:disjointWith>
    <owl:Class rdf:ID="Bacterium"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="MicroOrganism"/>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  >Virus</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Bacterium">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#MicroOrganism"/>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  >Bacterium</rdfs:label>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
  </rdfs:comment>
</owl:Class>
<owl:Class rdf:about="#MicroOrganism">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:ID="Organism"/>
        <owl:Restriction>
          <owl:someValuesFrom>
            <owl:Class rdf:ID="MicroScale"/>
          </owl:someValuesFrom>
          <owl:onProperty>
            <owl:ObjectProperty rdf:ID="asScaleRealm"/>
          </owl:onProperty>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

Tools are being developed for OWL



Even with nice XML tools, RDF syntax is not very nice to work with



Open world vs. Closed world



- Open world: A statement can be true unless explicitly known to be false.
 - What is not known/stated *can* be true.
- Closed world: A statement is false unless explicitly known to be true.
 - What is not known is assumed to be false.



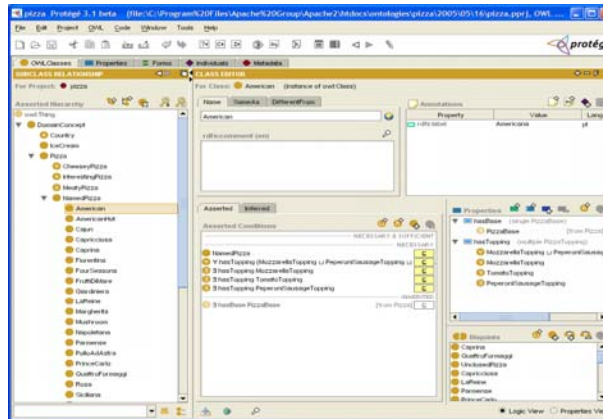
Basic Protégé-OWL usage

Nigam Shah
nigam@stanford.edu

Protégé OWL: a GUI environment



- OWL environment within PROTÉGÉ framework
- Most widely used tool for editing and managing OWL ontologies
- Approx 107,000 registered users



Protégé OWL features



- Loading and saving OWL files & databases
- Graphical editors for class expressions
- Access to description logics (DL) reasoners via Protégé GUI and the DIG interface
- Ontology visualization plug-ins
- Built on Protégé platform
 - Can hook in custom-tailored components
 - Can serve as API for new applications (including web applications)

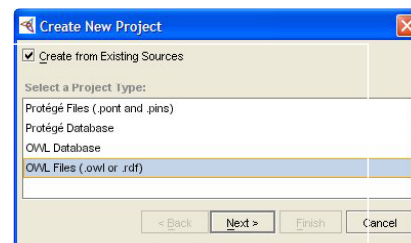


PROJECTS

Loading OWL files

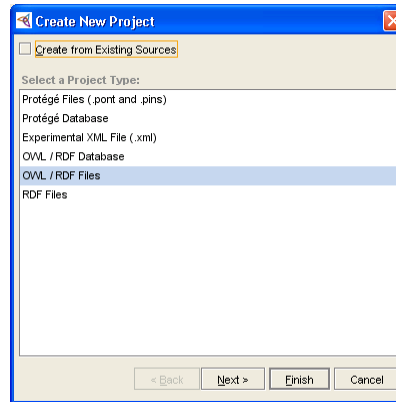


1. If you only have an OWL file:
 - **File** → **New Project**
 - Select **OWL Files** as the type
 - Tick **Create from existing sources**
 - **Next** to select the .owl file
2. If you've got a valid project file*:
 - **File** → **Open Project**
 - select the .pprj file



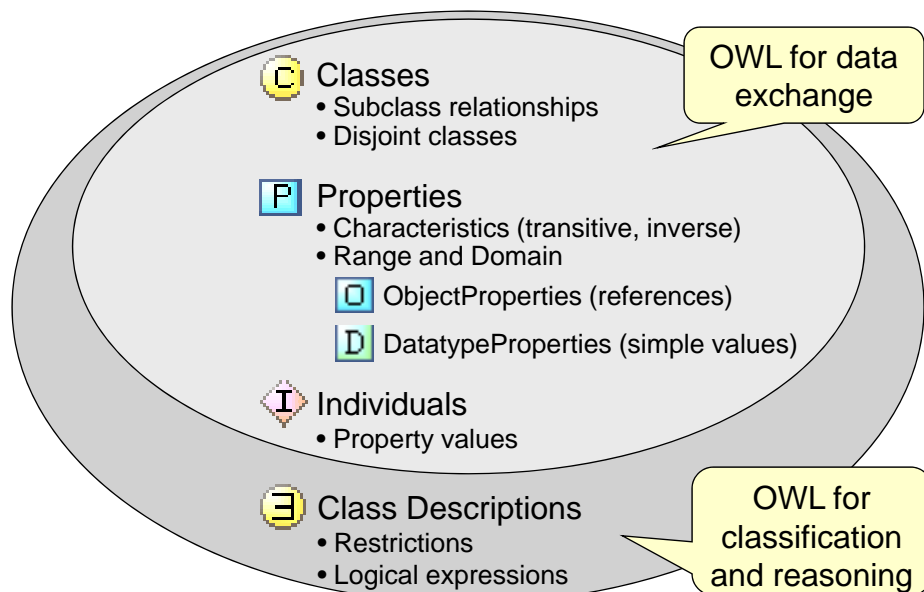
* ie one created on this version of Protégé - the s/w gets updated once every few days, so don't count on it unless you've created it recently– safest to build from the .owl file if in doubt

(Create or load an OWL project)

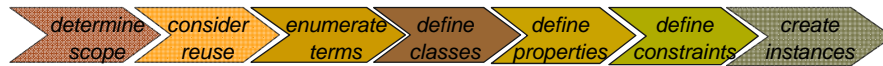


File → New Project
OR
File → Open Project

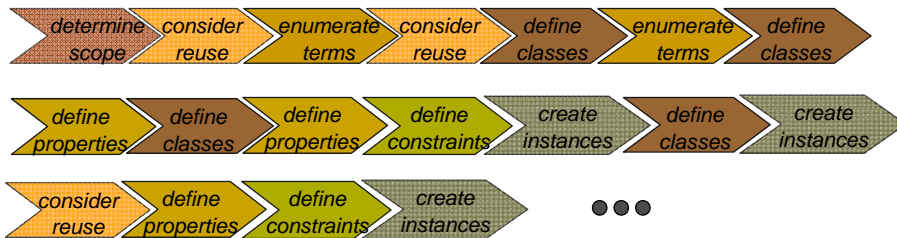
Protégé OWL Overview



Ontology Development Process



In reality - an iterative process:



Establish Purpose



What will the ontology be used for?

Classification of Pneumonia:

- Bacterial Pneumonia (caused by bacteria)
- Pneumococcal Pneumonia (caused by a particular kind of bacteria)
- Viral Pneumonia (caused by viruses)
- Mixed Pneumonia (caused by both bacteria and viruses)

Enumerate Important Entities



- What are the entities we need to talk about?
Pneumonias, infectious organisms.
- What are the properties of these entities?
hasRadiologyFinding, hasLocus, hasCause.
- What do we want to say about the entities?
Pneumonias cause radiology opacity findings
Pneumonias are located in lung
Mixed pneumonias are caused by bacteria and viruses.
...

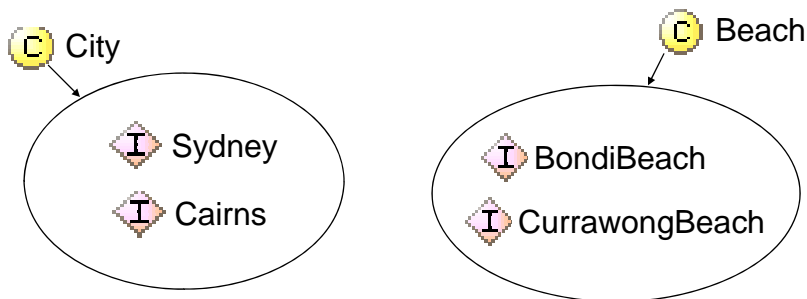


CLASSES (Types, Universals)

Classes



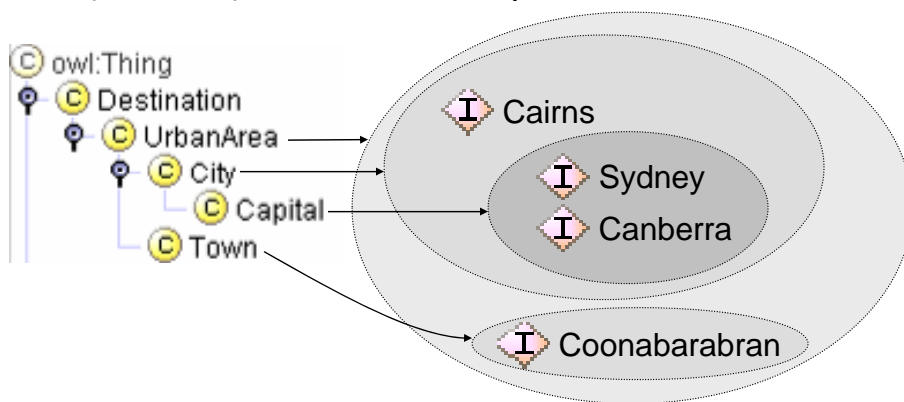
- Sets of **individuals** with common characteristics
- **Individuals** are *instances* of at least one class



Superclass Relationship



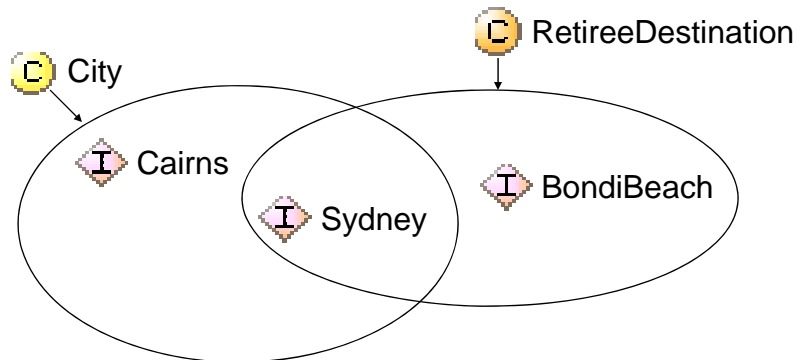
- Classes organized in a hierarchy implies subsumption
- Direct instances of subclass are also (indirect) instances of superclasses



Class overlap



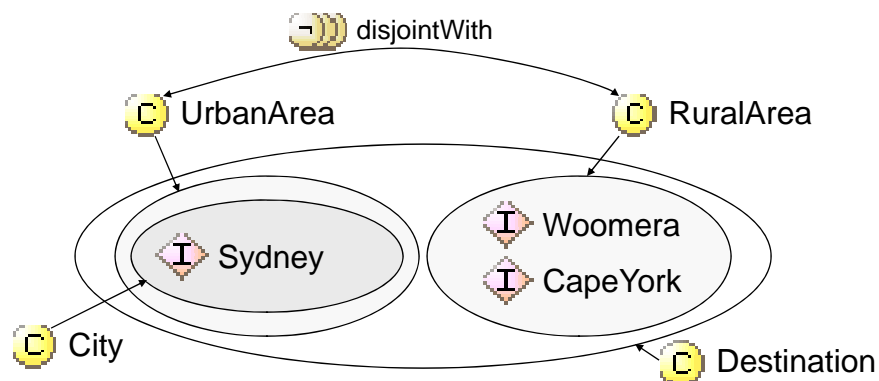
- Classes can overlap arbitrarily
- Classes are assumed non-disjoint by default (ie, they may share instances)



Class Disjointness



- All classes could potentially overlap
- Specify **disjointness** to make sure they don't share instances



Class Editor



Class annotations (for class metadata)

Class name and documentation

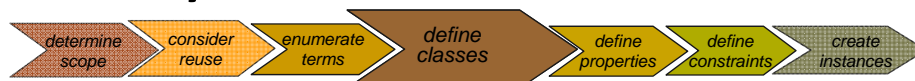
Properties "available" to Class

Disjoints widget

Conditions Widget

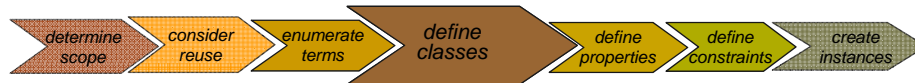
Class-specific tools (find usage etc)

Define classes and the class hierarchy



- Identify Classes (from the previous entity list)
 - If something can have a kind then it is a Class
 - "Kind of Pneumonia" √ - Pneumonia is a Class
 - "Kind of Samson" X - Samson is an individual
 - "Kind of Bacteria" √ Bacteria is a Class

Define classes and the class hierarchy



- Arrange Classes in an hierarchy
 - PneumococcalPneumonia is a subclass of Pneumonia
 - Every PneumococcalPneumonia is a Pneumonia
 - Pneumococcus is a subclass of Bacteria
 - Every Pneumococcus is a Bacteria
 - MixedPneumonia is a subclass of Pneumonia
 - Every MixedPneumonia is a Pneumonia

Create classes: “Pneumonia” class



The screenshot shows the Protégé 3.2 beta interface. The top menu bar includes File, Edit, Project, OWL, Code, Tools, Window, and Help. The toolbar contains various icons for file operations and editing. The main window is divided into two panes. The left pane, titled 'SUBCLASS EXPLORER', shows an 'Asserted Hierarchy' for the project 'TutorialTop-01'. The hierarchy starts with 'owl:Thing' and branches into 'DomainConcept', 'RefiningConcept', 'SelfStandingConcept', 'ActionRole', and 'Physical'. Under 'Physical', there is a 'PhysicalProcess' class, which has subclasses 'OrganicProcess' and 'MicroOrganicProcess'. 'OrganicProcess' has subclasses 'Disorder' and 'Pneumonia'. 'MicroOrganicProcess' has subclasses 'PhysicalStructure', 'InorganicStructure', 'MacroOrganicStructure', 'MacroStructure', 'MicroOrganicStructure', 'MicroStructure', 'OrganicStructure', 'PhysicalSubstance', and 'OrganicSubstance'. The right pane, titled 'CLASS EDITOR', shows the 'Pneumonia' class. It has a tab for 'Pneumonia' (instance of owl:Class) and an 'Inferred View' checkbox. The 'Asserted Conditions' section shows a table with the following conditions:

Condition	Relationship	Value
Disorder	NECESSARY & SUFFICIENT	NECESSARY
Disorder	NECESSARY	E
Disorder	INHERITED	[from OrganicProcess] E

The bottom of the interface has a status bar with various icons.

Class Disjoints



Note that Bacterial Pneumonia

- has superclass Pneumonia as a necessary condition
- Is asserted to be disjoint from its 'siblings'

The screenshot shows the Protégé 3.2 beta interface. On the left, the SUBCLASS EXPLORER displays a hierarchy: owl:Thing -> DomainConcept -> RefiningConcept -> SelfStandingConcept -> ActionRole -> Physical -> PhysicalProcess -> OrganicProcess -> Disorder -> Pneumonia -> BacterialPneumonia, ViralPneumonia, MixedPneumonia. On the right, the CLASS EDITOR for BacterialPneumonia shows its asserted conditions. A green arrow points to 'Pneumonia' under 'NECESSARY & SUFFICIENT', labeled 'Necessary parent'. Another green arrow points to the 'Disjoint classes' section, labeled 'Disjoint classes', which lists ViralPneumonia and MixedPneumonia.

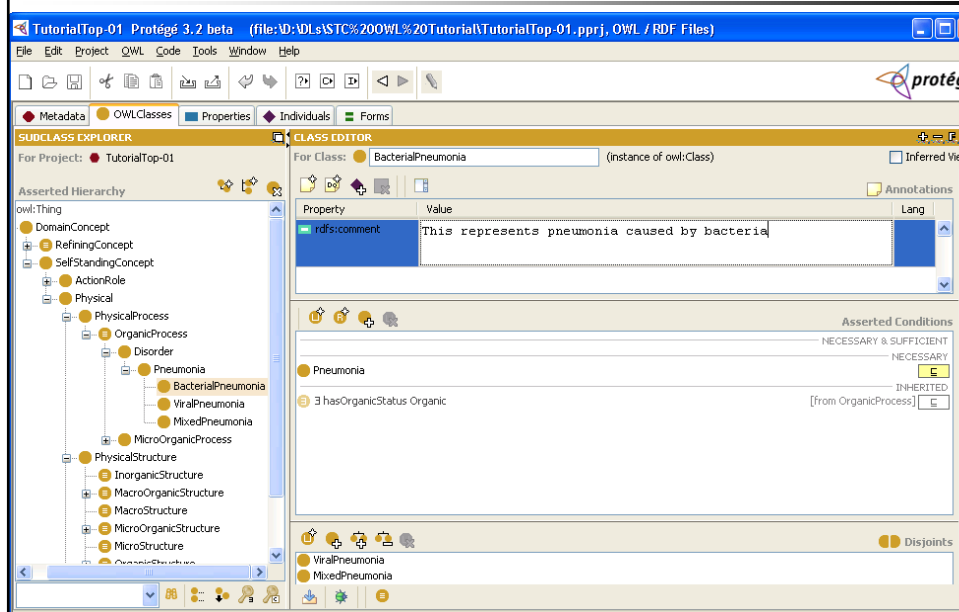
What it means



- All BacterialPneumonias are Pneumonias
 - **No BacterialPneumonia is not a Pneumonia**
- Nothing is both:
 - a BacterialPneumonia and a ViralPneumonia
 - a BacterialPneumonia and a MixedPneumonia

NB: In OWL classes *can overlap* unless declared disjoint!

Add metadata on Classes

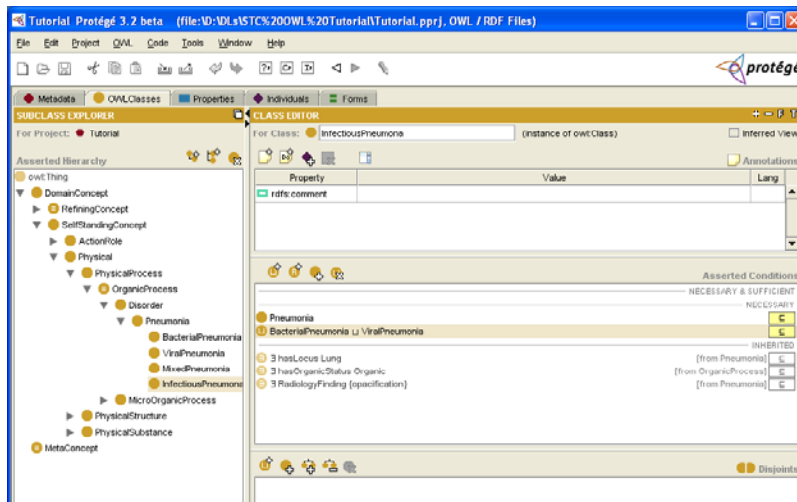


Another Way to Create Classes



- A class can be the **union** of two classes
 - An InfectiousPneumonia is either a BacterialPneumonia or a ViralPneumonia
- A class can be the **intersection** of two classes
 - A MixedPneumonia is any Pneumonia that is caused by both Bacteria and Viruses
- A class can be the **complement** of another class
 - Noninfectious pneumonia is any pneumonia that is not caused by an infectious agent (bacteria or virus)

Create a class by composition



An InfectiousPneumonia is a Pneumonia that is either a BacterialPneumonia or a ViralPneumonia



PROPERTIES

OWL Properties




- **Datatype Property** – relates Individuals to data (int, string, float etc)
 - [Pneumonia hasRadiologyFinding xsd:String](#)
- **Object Property** – relates Individuals
 - [BacterialPneumonia hasCause Bacterium](#)
- **Annotation Property** – for attaching metadata to classes, individuals or properties
 - [OntologyClass hasAuthor Natasha](#)

Datatype Properties



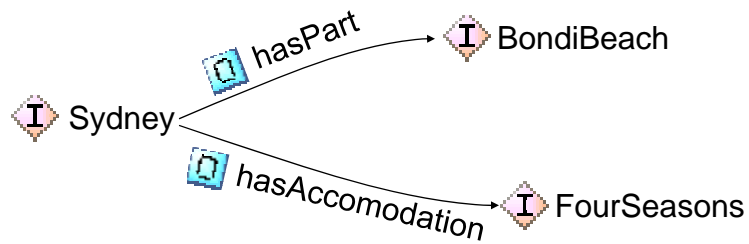
- Link individuals to primitive values (integers, floats, strings, booleans etc)
- Often: AnnotationProperties without formal “meaning”

 Sydney
hasSize = 4,500,000 isCapital = true rdfs:comment = “Don’t miss the opera house”

Object Properties



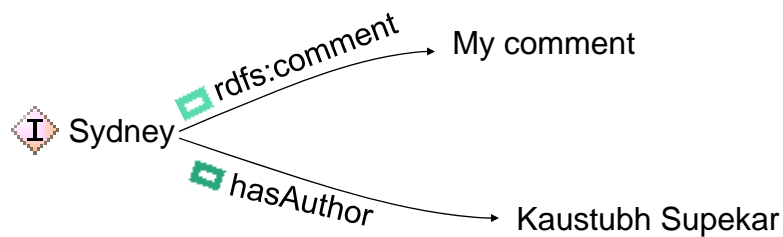
- Link two individuals together
- Relationships (0..n, n..m)



Annotation Properties



- To annotate classes, properties, and individuals
- Usually used for documentation



Mathematical properties of an OWL 'property'



- Functional
 - Person has_Mother Mother
- Transitive
 - A hasPart B, B hasPart C ==> A hasPart C
- InverseFunctional
 - Person has_SSN SSN
- Symmetric
 - A worksWith B ==> B worksWith A

Define Properties of Classes

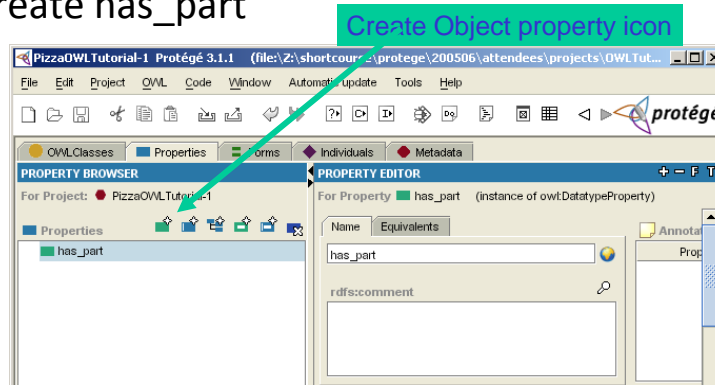


- Properties in a class definition describe attributes of instances of the class and relations to other instances
 - **Each Pneumonia will have radiology findings and a cause**
 - **Each cause for pneumonia will have a causative organism.**

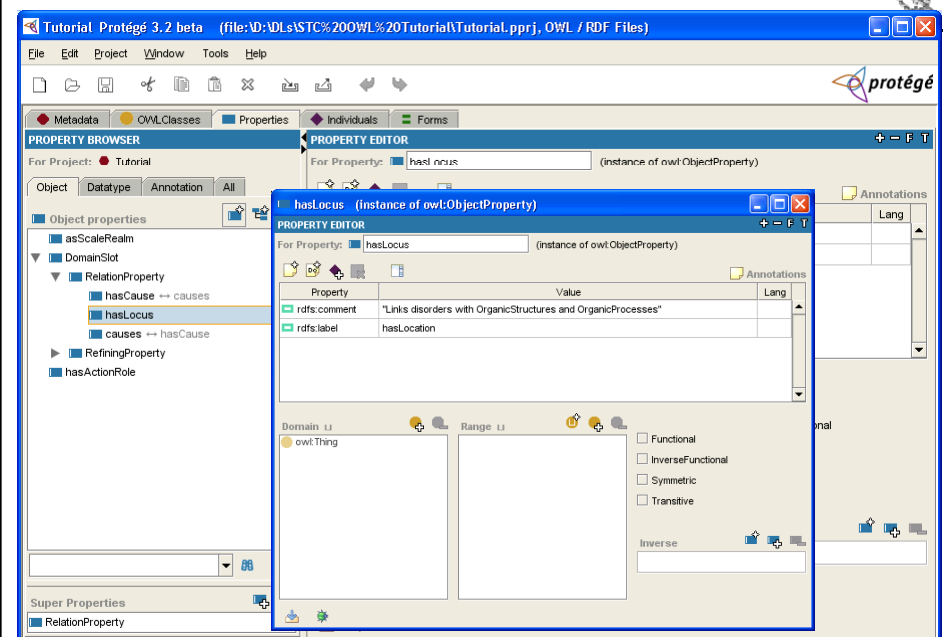
Create object property "has_part"



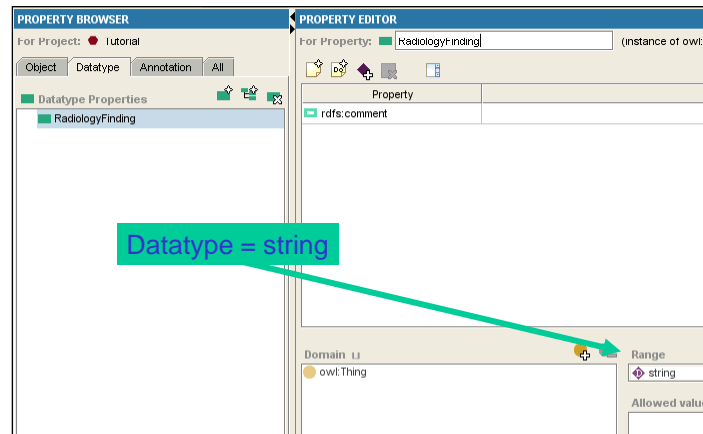
- Click on properties tab
- Click on Create_Object_property icon and create has_part



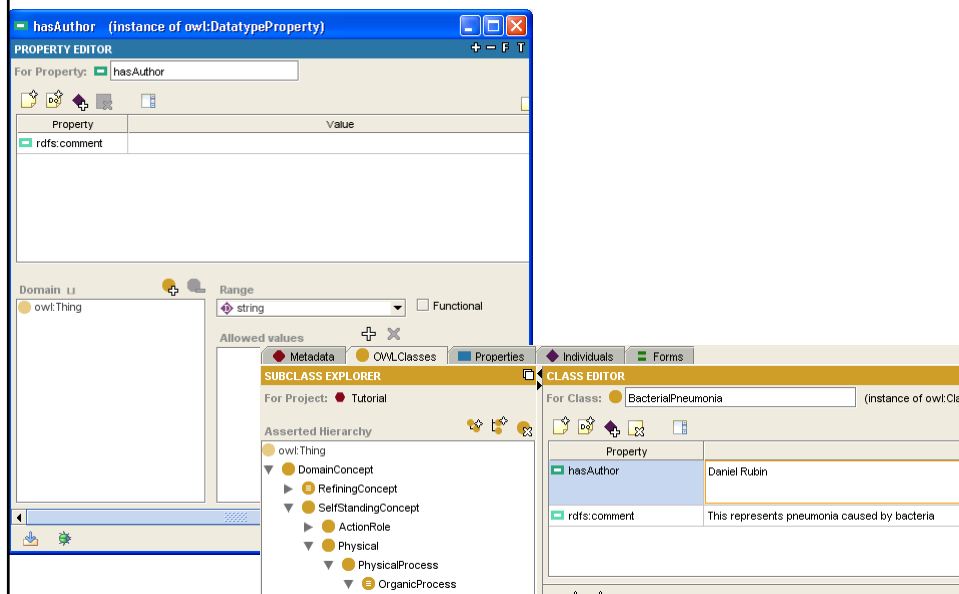
Object property hasLocus (already present)



Datatype Property “hasRadiologyFinding”



Create annotation property “hasAuthor”





RESTRICTIONS



Restrictions (Overview)

- An ***anonymous class*** consisting of all individuals that fulfill the condition
- Define a condition for property values
 - ⚡ allValuesFrom
 - ⚡ someValuesFrom
 - ⚡ hasValue
 - ⚡ minCardinality
 - ⚡ maxCardinality
 - ⚡ cardinality

Define Constraints : OWL Restrictions



- **Quantifier** restriction
 - How to represent the fact that every pneumonia must be located in a lung?
- **Cardinality** restrictions
 - How to represent that a lung must have 3 lobes as parts ?
- **hasValue** restrictions
 - How to define the value of a relation for a class ? (relationship between class and a individual)

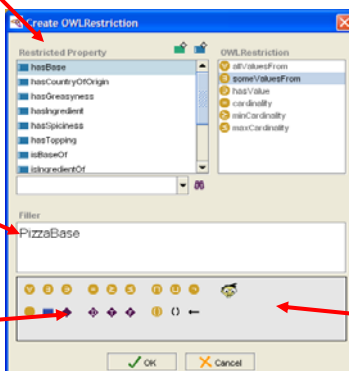
Creating Restrictions



Restricted Property

Filler Expression

Expression Construct Palette



Restriction Type

Syntax check

Create a restriction: using a datatype property



The screenshot shows the Protege 3.2 beta interface. On the left, the 'Create Restriction' dialog is open. In the 'Restricted Property' list, 'RadiologyFinding' is selected. In the 'Restriction' list, 'someValuesFrom' is selected. The 'Filler' text box contains 'opacification'. The 'OK' button is highlighted. On the right, the 'CLASS EDITOR' for 'Pneumonia' is shown. The 'Property' tab is active, and the 'Value' column shows '3 RadiologyFinding (Opacification)'. Below the dialog, a text box contains the following statement:

“All pneumonias are disorders that have a radiological finding of opacification”

Create a restriction: using an object property



The screenshot shows the Protege 3.2 beta interface. On the left, the 'Create Restriction' dialog is open. In the 'Restricted Property' list, 'hasLocus' is selected. In the 'Restriction' list, 'someValuesFrom' is selected. The 'Filler' text box contains 'Lung'. The 'OK' button is highlighted. On the right, the 'CLASS EDITOR' for 'Pneumonia' is shown. The 'Property' tab is active, and the 'Value' column shows '3 hasLocus Lung'. A green arrow points from a text box to the '3 hasLocus Lung' entry. Below the dialog, a text box contains the following statement:

“All pneumonias are disorders that are located in some lung and have a radiological finding of opacification”

... more object properties



- BacterialPneumonia is caused by some bacteria
 - $\text{BacterialPneumonia} \sqsubseteq \text{causedBy some Bacteria}$
 - $\text{BacterialPneumonia} \rightarrow \exists \text{ causedBy.Bacteria}$
- ViralPneumonia is caused by some virus
 - $\text{ViralPneumonia} \sqsubseteq \text{causedBy some Virus}$
- MixedPneumonia is caused by some bacteria and by some virus
 - $\text{MixedPneumonia} \sqsubseteq (\text{causedBy some Bacteria}) \sqcap (\text{causedBy some Virus})$



CLASS EXPRESSIONS

Using expression editor



“All MixedPneumonias are Pneumonias caused by Bacteria and by Viruses”

Class Descriptions



- Define the “meaning” of classes
- Description Logic expressions (“anonymous class expressions”) are used:
 - “All national parks have campgrounds.”
 - “A backpackers destination is a destination that has budget accommodation and offers sports or adventure activities.”
- **Expressions restrict property values**
- Reasoners can perform inference/classification

Defined/Primitive Classes



C Necessary Conditions:
(Primitive / partial classes)
“If we know that something is a X,
then it must fulfill the conditions...”

C Necessary & Sufficient Conditions:
(Defined / complete classes)
“If something fulfills the conditions...,
then it is an X.”

C NationalPark

NECESSARY & SUFFICIENT

NECESSARY

- RuralArea
- 3 hasAccommodation Campground
- 3 hasActivity Hiking

E E E

C QuietDestination

NECESSARY & SUFFICIENT

Destination

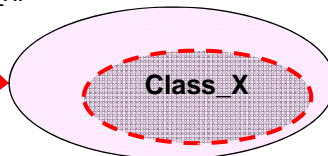
¬FamilyDestination

NECESSARY

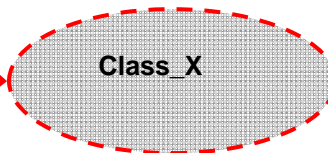
Defined/Primitive Classes



C Necessary Conditions: (Primitive classes)
Describes a subclass
“If something is a Class_X, then it must fulfill the conditions...”
Converse may NOT be true: “If something fulfills the conditions..., then it is a Class_X.”



C Necessary & Sufficient Conditions: (Defined classes)
“If something fulfills the conditions..., then it is a Class_X.”



e.g., Disorder is a *necessary* condition on Pneumonia



“If something is a Pneumonia, then it is a Disorder”
BUT
“If something is a Disorder, it may not be a Pneumonia”

Necessary & sufficient conditions on BacterialPneumonia



“If N&S conditions, then it is a BacterialPneumonia”
AND
“If something is a BacterialPneumonia, then N&S condtions”



INDIVIDUALS

Individuals



- Represent specific things in the domain
- Two names could represent the same “real-world” individual

 Sydney

 BondiBeach

 SydneysOlympicBeach

Create OWL instances



Create an instance of a class

- The class becomes a **direct type** of the instance
- Any superclass of the direct type is a **type** of the instance
- Generally, you create instances if you have a “type-of” something



Classification

Reasoners



- Reasoners (“classifiers”) infer information that is not explicitly contained within the ontology
- Standard reasoner services are:
 - **Consistency Checking** (i.e., satisfiability—can a class have any instances?)
 - **Subsumption Checking** (Finding subclasses—is A a subclass of B?)
 - **Equivalence Checking**
 - **Instantiation Checking** (Which classes does an individual belong to)
- For Protégé Pellet is pre-configured (but other tools with DIG support work too)
- Reasoners can be used at runtime in applications as a querying mechanism
- Used during development as an ontology “**compiler**”. Ontologies can be compiled to check if the meaning is what was intended

Run a DL Reasoner with Protégé OWL



- Protégé OWL can work with multiple reasoners
 - Racer (<http://www.racer-systems.com/>)
 - Pellet (<http://www.mindswap.org/2003/pellet/>)
 - Fact++ (<http://owl.man.ac.uk/factplusplus/>)
- Need to install, configure, and run at least one reasoner
 - Pellet comes pre-configured
- Protégé OWL and reasoner exchange information through inter-process communication



Common Mistakes

Common errors

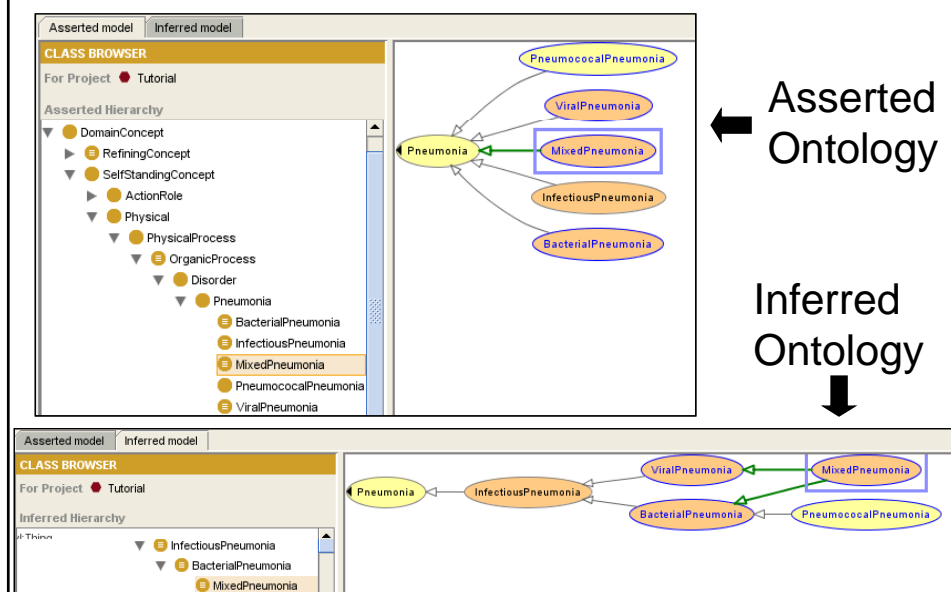


- **Lack of disjointness axioms causes what looks like "obvious misclassification"**
 - Open world assumption: classes can overlap unless you explicitly say they don't.
- **Lack of a closure axiom causes misclassification**
- **Trivial satisfaction of constraints (e.g. a missing existential 'some' declaration) can lead to weird classification**
 - e.g. an infection caused only by (herpes virus and HIV virus) will get classified as a bacterial infection!
 - ... because, the (herpes virus and HIV virus) is a null-set and the null-set is a subset of the 'bacteria' (which are disjoint with viruses) and hence the "an infection caused only by (herpes virus and HIV virus)" will be called a bacterial infection!
 - ONLY does not mean SOME



Visualization

Visualizing our OWL example





Limitations

What OWL can not do



- **OWL is a subset of first order logic with two variables**
 - “Binary relational”
 - n-ary relations require “reifying” the relation
 - i.e. re-representing the relation as a class
 - See n-ary relations note at W3C
 - n-ary predicates can always be represented as binary predicates
 - subject to other limitations of OWL
- **No representation for “same”**
 - The Owner is the same as the Person responsible
- **No representation for “All-All” statements**
 - “All licensed drivers are authorized to drive all cars”
 - Known to be tractable but no implementation available

What OWL can not do



- **Representation of “optional” properties is not standardized.**
Options:
 - Min cardinality 0
 - Member of the domain
- **No representation of uncertainty**
- **No representation of defaults and exceptions**
- **No higher order predicates**
 - OWL-DL is strictly first order
 - Can't say ... “Members of endangered species”
- **No closed world reasoning**
 - Everything must be closed explicitly
 - Take care when transforming from databases!



Summary

What does all this mean?



- Description logic (and OWL-DL) provides
 - Expressivity with semantic precision
 - Compositional definitions:
 - define new classes from old
 - Automatic classification & consistency checking
- Protégé OWL provides a GUI for developing OWL ontologies
- Be careful with
 - Open world reasoning
 - Use closure axioms when needed
 - “some” and “only” – someValuesFrom/allValuesFrom
 - domain and range constraints
 - making disjoint explicit



Further Reading

Assigned Reading



- Horridge, M. A Practical Guide To Building OWL Ontologies With the Protégé-OWL Plugin,
 - Edition 1.0, The University of Manchester, pp. 1-118, 2004.
 - <http://co-ode.man.ac.uk/resources/tutorials/ProtegeOWLTutorial.pdf>
- Knublauch, H. Weaving the Biomedical semantic web with protégé-owl.
 - <http://protege.stanford.edu/plugins/owl/publications/KRMed2004-protege-owl.pdf>
- Haimowitz IJ, Patil RS, and Szolovits P. Representing medical knowledge in a terminological language is difficult.
 - In: Proceedings of the Symposium on Computer Applications in Medical Care, IEEE Computer Society Press, pp. 101–105, 1988.
 - http://medg.lcs.mit.edu/people/psz/Haimowitz_NIKL.html

Further exploration



- **Protégé:** <http://protege.stanford.edu>
 - Protégé OWL:
<http://protege.stanford.edu/plugins/owl/>
 - Protégé OWL discussion list
 - Protégé Workshops (early 2006)
 - Protégé International Conference
- **OWL tutorial** materials from CO-ODE project site (University of Manchester)
<http://www.co-ode.org/resources/tutorials/>
- **NCBO:** <http://bioontology.org>

More about Protégé OWL



- Documentation on
<http://protege.stanford.edu/plugins/owl/documentation.html>
- Excellent tutorial by Mathew Horridge
<http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf>
- Other resources at <http://www.co-ode.org/resources/>